Connect - Python

User Guide

Prepared by Andrew McSween

Version 2.0

11/10/23

Table of Contents

1 About This Document

This document is the User Guide for the Connect - Python Capability. The document includes details on how to get started and the available requests. For additional details, please review the Jupyter Notebook samples on the ICE Connect Python website.

2 Getting Started!

2.1 Pre-Requisites

- Python - Version 3.4 or later
- ICE XL - Version 5.4 or later
- Python Library: pywin32 - Allows for the communication between Python and .Net Libraries
- Python Library: icepython - Connect Library for data requests

2.2 Installation

- Ensure Python 3.4 (or later) is properly installed on the Client machine
- Ensure the latest ICE XL version is installed, v. 5.4 or later
    - Installer found here
- Install pywin32 using pip command
    - Open command prompt
    - Type: python -m pip install --user pywin32
    - Note: "--user" is not needed if you're an admin with an admin installation of python
    - Confirm that the library has installed
- Install icepython library using pip command
    - From command prompt go to your ICE XL directory
    - Type: cd "%localappdata%\ICE Data Services\ICE XL\bin"
    - Now install the icepython library
    - Type: python -m pip install theice.com_ICEPython-0.0.6-py3-none-any.whl
    - Confirm that the library has installed

2.3 Importing the Library

- Within Python, import the icepython library
- Type: "import icepython as ice" from within Python
- Example:

```
In [2]: #Sample wrapper package import
        import icepython as ice # ICE Python wrapper
        import pandas as pd
        import matplotlib as mp
```

2.4 Additional Notes

- ICE XL must be installed and authenticated. The Connect - Python functionality uses ICE XL for authentication
- Any data limits or entitlements are shared between ICE XL and Connect - Python
- All Python requests must happen locally to the machine running ICE XL

- The Python library is installed and updated through the ICE XL installer

3 Snapshot Quotes

3.1 Snapshot Get Quotes

- Function: **Get_quotes**(symbols, fields, bool subscribe=false)
- Snapshot quotes returns the latest value for streaming content
- Number of symbols is limited to 500 per request
- Number of requests are limited to 10 per second

Arguments

- Request takes a set of symbols, a set of fields, and an additional argument to keep the data streaming to the publisher
- Symobls and Fields arguments are required
    - Symbols can be set as an array
-
```
In [6]: symbols = ['IBM', 'GOOG', 'ICE', 'AMD']
        print(symbols)

        ['IBM', 'GOOG', 'ICE', 'AMD']
```
    - Fields can be set as an array
-
```
In [7]: fields = ['bid', 'ask', 'last', 'volume']
        print(fields)

        ['bid', 'ask', 'last', 'volume']
```
    - subscribe = True will set the subscription to be maintained on the publisher side so subsequent requests are faster - the latest updates are local. It is recommended that you subscribe to regularly used symbols.
    - subscribe = False will unsubscribe the subscription after the initial request. Future requests will require re-subscribing. This will result in slower updateds

Example

```
In [29]: #Sample Quote request
         my_data = icexl.get_quotes(symbols, #List of symbols
                                    fields, #List of fields
                                    True) #True keeps the connection to those symbols open, so subsequent requests are from a local cache
```

3.2 Snapshot Time Series Details

- Function: **get_timeseries**(symbols, fields, granularity, start_date, end_date, timeline, timeref, intradayfilter)
- Time series request returns a tuple where the date is in the first column, followed by the symbols and fields requested
- If multiple fields are requested, the output is sorted so all fields for one symbol are placed together
- Output has column headers with the symbols and fields, e.g. “IBM.Last”

Arguments

- Symbols, Fields, Granularity, Start_date, End_date are required fields
- Symbols can be set as an array
- Fields can be set as an array
- Granularity is one value
    - D = Daily
    - W = Weekly
    - M = Monthly
    - I1 = 1 min
    - I30 = 30 min
    - I60 = 1 hour
- Start_date can take a date, or date & time
- End_date can take a date or date & time
- Timeline (optional)
- Timeref (optional)
- Intradayfilter (optional) - This argument will filter results such that you only return rows that occur within the provide time filter

Example of basic timeseries request

```
In [ ]: #Sample Time Series Request
        my_data = ice.get_timeseries(symbols, #Set of symbols in the request
                                     fields, #Set of feilds in the request
                                     granularity='D', #defines the timeseries granularity
                                     start_date='2020-01-01', #Start date
                                     end_date='2020-12-31') #End date
```

Example of timeseries request filtered on only times between 11:11 and 19:00

```
In [4]: a=ice.get_timeseries('BRN 1!-ICE', 'Last',
                             granularity='I1',
                             start_date='2023-09-28',
                             end_date='2023-09-29',timeline='',
                             intradayfilter='11:11-19:00')
        for l in a:
            print(l)
```

```
('Time', 'BRN 1!-ICE.Last')
('2023-09-28T12:11:00', 91.81)
('2023-09-28T12:12:00', 91.86)
('2023-09-28T12:13:00', 91.92)
('2023-09-28T12:14:00', 91.93)
('2023-09-28T12:15:00', 91.93)
('2023-09-28T12:16:00', 91.91)
('2023-09-28T12:17:00', 91.92)
```

3.3 Snapshot Time & Sales Details

Function: **get_timesales**(symbols, fields, start_date, end_date, filter, conditions, conditionslogic)

- Snapshot Time & Sales requests are limited to 1 per second
- Snapshot Time & Sales requests are limited to 10 symbols per request
- Data is returned in a tuple with date and time in the left most column, with symbols and fields listed in the headers of the columns to the right
- If multiple fields are requested, the output is sorted so all fields for one symbol are placed together

Arguments

- Symbols can be set as an array
- Fields can be set as an array
- Start_date can take a date, or date & time
- End_date can take a date or date & time
- Optional Filter argument can filter results on available filters. See get_filters() to pull available filters
- Optional conditions array applies a set of conditions
- Optional conditionslogic applies a logical operator to your set of conditions (eg, AND, OR)

Example: Basic Time and Sales query

```
In [ ]: #Sample Time & Sales Request
        my_data = ice.get_timesales(symbols, #Set of symbols in the request
                                    fields, #Set of feilds in the request
                                    start_date='2020-12-30T15:30:00', #Start date and time
                                    end_date='2020-12-31T15:30:00') #End date and time
```

Example: Time and Sales filtered on Blocks

```
import pandas as pd
data = ice.get_timesales('HNG 1!-IUS',
                         ['Price','Conditions'],
                         '2023-10-10','2023-10-11',
                         filter='Blocks')
df = pd.DataFrame(list(data))
print(df)
```

```
                     0                 1                      2
0                 Time  HNG 1!-IUS.Price  HNG 1!-IUS.Conditions
1  2023-10-10T11:37:34             3.635              BlockTrde
2  2023-10-10T12:53:42             3.635              BlockTrde
3  2023-10-10T14:11:45              3.65              BlockTrde
4  2023-10-10T14:26:42              3.64         BlockTrde, Leg
5  2023-10-10T15:00:52              3.66              BlockTrde
6  2023-10-10T15:06:15             3.649         BlockTrde, Leg
7  2023-10-10T15:07:56             3.707         BlockTrde, Leg
8  2023-10-10T16:02:08              3.66         BlockTrde, Leg
9  2023-10-10T16:02:34              3.66         BlockTrde, Leg
```

Example: Time and Sales filtered on multiple Conditions, applying AND to the conditions list.

```
import pandas as pd
data = ice.get_timesales('HNG 1!-IUS',
                         ['Price','Conditions'],
                         '2023-10-10','2023-10-11',
                         conditions=['BlockTrde', 'Leg'],
                         conditionslogic='AND')
df = pd.DataFrame(list(data))
print(df)
```

```
                    0                 1                      2
0                Time  HNG 1!-IUS.Price  HNG 1!-IUS.Conditions
1  2023-10-10T14:26:42              3.64         BlockTrde, Leg
2  2023-10-10T15:06:15             3.649         BlockTrde, Leg
3  2023-10-10T15:07:56             3.707         BlockTrde, Leg
4  2023-10-10T16:02:08              3.66         BlockTrde, Leg
5  2023-10-10T16:02:34              3.66         BlockTrde, Leg
```

4 Metadata, Symbol and Field Lookup Requests

4.1 Autolisting function

- Function: get_autolist(root symbol)
- Autolisting allows for the request of the set of tenors of a given product
- Results should be returned in an array, which can be used in future requests
- Monthly futures can be requested with one "*" and the product "Root"
  - "*BRN-ICE" will return all futures
- Spreads can be requested with two "*" and the product "Root"
  - "**BRN-ICE" will return front-back calendar spreads
  - "**BRN:BRN-ICE" will return all calendar spreads
  - "**UHU:BRN-ICE'" will return intercommodity spreads
- Historical data can be requested with + operator
  - "*+BRN-ICE'" will return historical tenors
- Options
  - "***BRN-ICE" will return all expirations for a given product/security
  - Use suffix "fut" when exchange suffix is not present.
    - Example: "symbols = ice.get_autolist('***sb fut')"
  - On Stocks, when exchange suffix is not present, add 'stk'
    - Example: symbols ="ice.get_autolist('***sb stk')"
  - "***BRN-ICE Z24" to get all options for provided expiration
  - "***BRN-ICE M22 ATM:10" to get all at the money options for a given expiration

- “***BRN-ICE Z24 OTM:10” to get all out of the money options for a given expiration
- “***BRN-ICE m22 ITM:10” to get all in the money options for a given expiration

Example: get intercommodity spread symbols and generate a times series

```
#Sample time series
symbols = ice.get_autolist('**UHU:BRN-ICE')
fields = 'Last'
data = ice.get_timeseries(symbols,
                          fields,
                          granularity = 'D',
                          start_date='2020-12-01',
                          end_date='2020-12-31',
                          )
print(data)
```

4.2 Get Conditions

- Function: get_conditions()
- Returns a list of all conditions

4.3 Get Quote Fields

- Function: get_quotes_fields()
- Returns fields available for symbols
- Note that this does not take a symbol argument

4.4 Get timeseries Fields

- Function: get_timeseries_fields(symbol)
- Returns all fields available for a specific symbol via the get_timeseries function

Example

```
In [12]: ice.get_timeseries_fields('IBM')

Out[12]: (('Open', 'Open'),
          ('High', 'High'),
          ('Low', 'Low'),
          ('Close', 'Close'),
          ('Last', 'Last'),
          ('Settle', 'Settle'),
          ('Volume', 'Volume'),
          ('Open Interest', 'Open Interest'),
          ('Unadjusted Open', 'Unadjusted Open'),
          ('Unadjusted High', 'Unadjusted High'),
          ('Unadjusted Low', 'Unadjusted Low'),
          ('Unadjusted Close', 'Unadjusted Close'),
          ('Unadjusted Last', 'Unadjusted Last'),
          ('Unadjusted Volume', 'Unadjusted Volume'),
          ('IM Bid Open', 'IM Bid Open'),
          ('IM Bid High', 'IM Bid High'),
          ('IM Bid Low', 'IM Bid Low'),
          ('IM Bid Close', 'IM Bid Close'),
          ('IM Ask Open', 'IM Ask Open'),
          ('IM Ask High', 'IM Ask High'),
          ('IM Ask Low', 'IM Ask Low'),
          ('IM Ask Close', 'IM Ask Close'),
          ('IM Trade Open', 'IM Trade Open'),
          ('IM Trade High', 'IM Trade High'),
          ('IM Trade Low', 'IM Trade Low'),
          ('IM Trade Close', 'IM Trade Close'))
```

4.5 Get Time and Sales Fields

- Function: get_timesales_fields(symbol)
- Returns all fields available for a specific symbol via the get_timesales() function

4.6 Get Deltas

- Function: get_deltas(fields)
- get_deltas takes a list of field names and returns a list of subscribed symbols where one of the fields has changed since the last update reset
- get_deltas resets after calling get_quotes or calling get_deltas with reset argument = True
- On an update reset, all fields and symbols are reset regardless of what was requested in the get_quotes or get_deltas call that initiated the reset.

Arguments

- List of fields to check for updates
- Optional bool reset=false. Setting to True will reset get_deltas

Example: Retrieve updates on only symbols that have been updated.

```
#sample request for retrieving list of subscribed symbols that have been updated since last time get_quotes was called
symbols = ['ibm', 'eur a0-fx', 'gme', 'ice', 'HNG 1!-IUS', 'BRN 1!-ICE']
fields = ['open','high','low','close','bid']
#subscribe to a set of symbols via get_quotes
data = ice.get_quotes(symbols,fields, True)
#return a list of symbols where a provided field has updated since get_quotes was last called.
symbols = ice.get_deltas(fields, False)
print (symbols)
```

```
['ibm', 'eur a0-fx', 'gme', 'ice', 'HNG 1!-IUS', 'BRN 1!-ICE']
()
```

```
#call get_quotes and retrieve only updated symbols using get_deltas function to filter down the symbol list
data = ice.get_quotes(ice.get_deltas(fields, False),fields, True)
print (data)
```

```
(('', 'open', 'high', 'low', 'close', 'bid'), ('IBM', 149.25, 149.68, 147.585, None, 147.7), ('EUR A0-FX', 1.06986, 1.0716, 1.0
6595, None, 1.07083), ('ICE', 108.31, 109.36, 107.76, None, 109.2), ('BRN 1!-ICE', 81.4, 81.96, 79.2, None, 79.84))
```

Arguments

- Single symbol

4.7 Get Search Facets

- Function: get_search_facets()
- Returns available search facets

4.8 Get Search Filters

- Function: get_search_filters(array of facets)
- Returns a set of filters based on provided facet

Arguments

- Array of facets

Example: Get filters based on 'types' facet

```
#See other filters based on facets, above
filters = ice.get_search_filters('types')
filters
```

```
(('F.TYP', 'Types', 1),
 ('F.TYP.6', 'Fundamental', 2),
 ('F.TYP.7', 'Bond', 2),
 ('F.TYP.3', 'Spread', 2),
 ('F.TYP.1', 'Future Option', 2),
 ('F.TYP.8', 'Stock', 2),
 ('F.TYP.9', 'Stock Option', 2),
 ('F.TYP.2', 'Cash Energy', 2),
 ('F.TYP.0', 'Future', 2),
 ('F.TYP.10', 'Assessment', 2),
 ('F.TYP.4', 'Index', 2),
 ('F.TYP.11', 'Fund', 2),
 ('F.TYP.5', 'Forex', 2),
 ('F.TYP.12', 'ETF', 2),
 ('F.TYP.13', 'Treasury', 2))
```

4.9 Get Search

- Function: get_search(symbol, rows=100, filters=[], bool symbolsOnly = False)
- Returns an array of symbols that match the search criteria. SymbolsOnly bool will return only the symbols rather than more detailed symbol data

Arguments

- Symbol
- Rows=n where n is the number of rows to return
- Filters=[array of filters]
- Bool symbolsOnly=False. Set to True to show only symbol name.

Example: Search on keyword and filters

```
In [22]: #Search on keyword and filters
         results = ice.get_search('brn', rows=10, filters='F.EX.7.11')
         results

Out[22]: (('BRN 24F-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24G-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24H-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24J-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24K-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24M-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24N-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24Q-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24U-ICE', 'Brent Crude Oil', 'ICE', 'Future'),
          ('BRN 24V-ICE', 'Brent Crude Oil', 'ICE', 'Future'))
```

5 Ancillary Requests

5.1 Help

- Function: Help
- Help(ice) will output available ICE Python functions

5.2 Hibernation functions

- Function: **get_hybernation**()
- The ICE XL Publisher will hibernate when not in use.
- Connection will occur as soon as a request is made and take the ICE XL Publisher out of hibernation mode
    - Connecting prior to the first request will speed up initial requests to the server
- Any active subscriptions will prevent the ICE XL Publisher from entering hibernation mode.
- There are two functions relating to Hibernation
    - get_hibernation() returns the current state of the publisher
    - set_hibernation() sets the ICE Publisher to a desired state

i Arguments

- Set_hibernation(VAL) can take **True** or **False** values

Examples:

```
In [15]: ice.get_hibernation()
Out[15]: 'False'
```

```
In [25]: ice.set_hibernation(True)
```

5.3 Start Publisher

- Function: start_publisher()

- Force publisher to start

5.4 Restart Publisher

- Function: restart_publisher()
- This will restart the ICEXL and Python publisher.

5.5 Get Subscriptions

- Function: get_subscriptions()
- Returns an array of symbols that you are currently subscribed to.
- Subscriptions can be assigned via the get_quotes() function with the subscribe argument = True.

```
In [11]: ice.get_subscriptions()
Out[11]: ('IBM', 'EUR A0-FX', 'GME', 'ICE', 'HNG 1!-IUS', 'BRN 1!-ICE')
```

5.6 Clear Subscriptions

- Function: clear_subscriptions()
- Removes all subscriptions

5.7 Set Timeout

- Function: set_timeout(timeout)
- Sets the timeout before the publisher sleeps

Arguments

- Timeout (in seconds)

6 Usage Guidelines

ICE Python usage guidelines are outlined below. Please contact us if these guidelines don't match your needs, ICE Data Services provides multiple ways to access datasets either via desktop applications or feeds. For more information, please consult your Data Services Agreement: https://www.ice.com/publicdocs/agreements/ICE_Data_Services_Agreement.pdf

6.1 ICE Python Request limits

- Get_quotes request is limited to 500 symbols per request
- Get_quotes requests are limited to 10 per second
- Snapshot Time & Sales requests are limited to 1 per second
- Snapshot Time & Sales requests are limited to 10 symbols per request